

Contents

1	Computing with formulas	1
1.1	The first programming encounter: a formula	2
1.1.1	Using a program as a calculator	2
1.1.2	About programs and programming	3
1.1.3	Tools for writing programs	3
1.1.4	Writing and running your first Python program	4
1.1.5	Warning about typing program text	5
1.1.6	Verifying the result	6
1.1.7	Using variables	6
1.1.8	Names of variables	7
1.1.9	Reserved words in Python	8
1.1.10	Comments	9
1.1.11	Formatting text and numbers	10
1.2	Computer science glossary	13
1.3	Another formula: Celsius-Fahrenheit conversion	18
1.3.1	Potential error: integer division	18
1.3.2	Objects in Python	19
1.3.3	Avoiding integer division	20
1.3.4	Arithmetic operators and precedence	21
1.4	Evaluating standard mathematical functions	22
1.4.1	Example: Using the square root function	22
1.4.2	Example: Computing with $\sinh x$	24
1.4.3	A first glimpse of round-off errors	25
1.5	Interactive computing	26
1.5.1	Using the Python shell	27
1.5.2	Type conversion	27
1.5.3	IPython	28
1.6	Complex numbers	31
1.6.1	Complex arithmetics in Python	32

1.6.2	Complex functions in Python	33
1.6.3	Unified treatment of complex and real functions	33
1.7	Symbolic computing	35
1.7.1	Basic differentiation and integration	35
1.7.2	Equation solving and Taylor series	36
1.8	Summary	37
1.8.1	Chapter topics	37
1.8.2	Example: Trajectory of a ball	41
1.8.3	About typesetting conventions in this book	42
1.9	Exercises	43
2	Loops and lists	53
2.1	While loops	53
2.1.1	A naive solution	53
2.1.2	While loops	54
2.1.3	Boolean expressions	56
2.1.4	Loop implementation of a sum	58
2.2	Lists	60
2.2.1	Basic list operations	60
2.2.2	For loops	63
2.3	Alternative implementations with lists and loops	64
2.3.1	While loop implementation of a for loop	65
2.3.2	The range construction	65
2.3.3	For loops with list indices	66
2.3.4	Changing list elements	67
2.3.5	List comprehension	68
2.3.6	Traversing multiple lists simultaneously	68
2.4	Nested lists	69
2.4.1	A table as a list of rows or columns	69
2.4.2	Printing objects	71
2.4.3	Extracting sublists	72
2.4.4	Traversing nested lists	74
2.5	Tuples	76
2.6	Summary	77
2.6.1	Chapter topics	77
2.6.2	Example: Analyzing list data	80
2.6.3	How to find more Python information	82
2.7	Exercises	84
3	Functions and branching	93
3.1	Functions	93
3.1.1	Mathematical functions as Python functions	93
3.1.2	Understanding the program flow	95
3.1.3	Local and global variables	96
3.1.4	Multiple arguments	98

- 3.1.5 Function argument of global variable? 99
 - 3.1.6 Beyond mathematical functions 100
 - 3.1.7 Multiple return values 101
 - 3.1.8 Computing sums 102
 - 3.1.9 Functions with no return values 103
 - 3.1.10 Keyword arguments 105
 - 3.1.11 Doc strings 107
 - 3.1.12 Functions as arguments to functions 109
 - 3.1.13 The main program 111
 - 3.1.14 Lambda functions 111
 - 3.2 Branching 112
 - 3.2.1 If-else blocks 113
 - 3.2.2 Inline if tests 114
 - 3.3 Mixing loops, branching, and functions in bioinformatics examples 115
 - 3.3.1 Counting letters in DNA strings 116
 - 3.3.2 Efficiency assessment 119
 - 3.3.3 Verifying the implementations 121
 - 3.4 Summary 122
 - 3.4.1 Chapter topics 122
 - 3.4.2 Example: Numerical integration 124
 - 3.5 Exercises 128
- 4 User input and error handling 147
 - 4.1 Asking questions and reading answers 148
 - 4.1.1 Reading keyboard input 148
 - 4.2 Reading from the command line 149
 - 4.2.1 Providing input on the command line 149
 - 4.2.2 A variable number of command-line arguments .. 150
 - 4.2.3 More on command-line arguments 151
 - 4.3 Turning user text into live objects 152
 - 4.3.1 The magic eval function 152
 - 4.3.2 The magic exec function 156
 - 4.3.3 Turning string expressions into functions 158
 - 4.4 Option-value pairs on the command line 159
 - 4.4.1 Basic usage of the argparse module 160
 - 4.4.2 Mathematical expressions as values 161
 - 4.5 Reading data from file 163
 - 4.5.1 Reading a file line by line 164
 - 4.5.2 Alternative ways of reading a file 165
 - 4.5.3 Reading a mixture of text and numbers 167
 - 4.6 Writing data to file 169
 - 4.6.1 Example: Writing a table to file 169
 - 4.6.2 Standard input and output as file objects 171
 - 4.6.3 What is a file, really? 173

4.7	Handling errors	177
4.7.1	Exception handling	178
4.7.2	Raising exceptions	181
4.8	A glimpse of graphical user interfaces	183
4.9	Making modules	186
4.9.1	Example: Interest on bank deposits	187
4.9.2	Collecting functions in a module file	188
4.9.3	Test block	188
4.9.4	Verification of the module code	190
4.9.5	Getting input data	191
4.9.6	Doc strings in modules	194
4.9.7	Using modules	194
4.9.8	Distributing modules	197
4.9.9	Making software available on the Internet	198
4.10	Summary	199
4.10.1	Chapter topics	199
4.10.2	Example: Bisection root finding	203
4.11	Exercises	211
5	Array computing and curve plotting	221
5.1	Vectors	222
5.1.1	The vector concept	222
5.1.2	Mathematical operations on vectors	224
5.1.3	Vector arithmetics and vector functions	225
5.2	Arrays in Python programs	227
5.2.1	Using lists for collecting function data	227
5.2.2	Basics of numerical Python arrays	228
5.2.3	Computing coordinates and function values	230
5.2.4	Vectorization	231
5.3	Curve plotting	233
5.3.1	Matplotlib; pylab	233
5.3.2	Matplotlib; pyplot	237
5.3.3	SciTools and Easyviz	239
5.3.4	Making animations	244
5.3.5	Making videos	249
5.3.6	Curve plots in pure text	251
5.4	Plotting difficulties	252
5.4.1	Piecewisely defined functions	252
5.4.2	Rapidly varying functions	255
5.5	More advanced vectorization of functions	256
5.5.1	Vectorization of StringFunction objects	256
5.5.2	Vectorization of the Heaviside function	257
5.5.3	Vectorization of a hat function	260
5.6	More on numerical Python arrays	262
5.6.1	Copying arrays	263

5.6.2	In-place arithmetics	263
5.6.3	Allocating arrays	264
5.6.4	Generalized indexing	264
5.6.5	Testing for the array type	265
5.6.6	Compact syntax for array generation	266
5.6.7	Shape manipulation	266
5.7	Higher-dimensional arrays	267
5.7.1	Matrices and arrays	267
5.7.2	Two-dimensional numerical Python arrays	268
5.7.3	Array computing	271
5.7.4	Two-dimensional arrays and functions of two variables	272
5.7.5	Matrix objects	272
5.8	Summary	274
5.8.1	Chapter topics	274
5.8.2	Example: Animating a function	275
5.9	Exercises	280
6	Dictionaries and Strings	301
6.1	Dictionaries	302
6.1.1	Making dictionaries	302
6.1.2	Dictionary operations	303
6.1.3	Example: Polynomials as dictionaries	304
6.1.4	Dictionaries with default values and ordering	306
6.1.5	Example: File data in dictionaries	309
6.1.6	Example: File data in nested dictionaries	310
6.1.7	Example: Reading and plotting data recorded at specific dates	314
6.2	Strings	318
6.2.1	Common operations on strings	318
6.2.2	Example: Reading pairs of numbers	322
6.2.3	Example: Reading coordinates	325
6.3	Reading data from web pages	327
6.3.1	About web pages	327
6.3.2	How to access web pages in programs	329
6.3.3	Example: Reading pure text files	329
6.3.4	Example: Extracting data from HTML	331
6.3.5	Handling non-English text	332
6.4	Reading and writing spreadsheet files	335
6.4.1	CSV files	335
6.4.2	Reading CSV files	336
6.4.3	Processing spreadsheet data	337
6.4.4	Writing CSV files	338
6.4.5	Representing number cells with Numerical Python arrays	339

6.4.6	Using more high-level Numerical Python functionality	340
6.5	Examples from analyzing DNA	341
6.5.1	Computing frequencies	341
6.5.2	Analyzing the frequency matrix	348
6.5.3	Finding base frequencies	351
6.5.4	Translating genes into proteins	353
6.5.5	Some humans can drink milk, while others cannot	358
6.6	Summary	359
6.6.1	Chapter topics	359
6.6.2	Example: A file database	361
6.7	Exercises	364
7	Introduction to classes	373
7.1	Simple function classes	374
7.1.1	Challenge: functions with parameters	374
7.1.2	Representing a function as a class	376
7.1.3	Another function class example	382
7.1.4	Alternative function class implementations	383
7.1.5	Making classes without the class construct	385
7.2	More examples on classes	387
7.2.1	Bank accounts	387
7.2.2	Phone book	389
7.2.3	A circle	391
7.3	Special methods	393
7.3.1	The call special method	393
7.3.2	Example: Automagic differentiation	394
7.3.3	Example: Automagic integration	397
7.3.4	Turning an instance into a string	399
7.3.5	Example: Phone book with special methods	400
7.3.6	Adding objects	402
7.3.7	Example: Class for polynomials	402
7.3.8	Arithmetic operations and other special methods	407
7.3.9	Special methods for string conversion	408
7.4	Example: Class for vectors in the plane	410
7.4.1	Some mathematical operations on vectors	410
7.4.2	Implementation	410
7.4.3	Usage	412
7.5	Example: Class for complex numbers	413
7.5.1	Implementation	414
7.5.2	Illegal operations	415
7.5.3	Mixing complex and real numbers	416
7.5.4	Dynamic, static, strong, weak, and duck typing ..	417
7.5.5	Special methods for “right” operands	419
7.5.6	Inspecting instances	420

7.6	Static methods and attributes	421
7.7	Summary	423
7.7.1	Chapter topics	423
7.7.2	Example: Interval arithmetics	424
7.8	Exercises	429
8	Random numbers and simple games	447
8.1	Drawing random numbers	448
8.1.1	The seed	448
8.1.2	Uniformly distributed random numbers	449
8.1.3	Visualizing the distribution	450
8.1.4	Vectorized drawing of random numbers	451
8.1.5	Computing the mean and standard deviation	452
8.1.6	The Gaussian or normal distribution	454
8.2	Drawing integers	455
8.2.1	Random integer functions	455
8.2.2	Example: Throwing a die	456
8.2.3	Drawing a random element from a list	459
8.2.4	Example: Drawing cards from a deck	459
8.2.5	Example: Class implementation of a deck	461
8.3	Computing probabilities	464
8.3.1	Principles of Monte Carlo simulation	464
8.3.2	Example: Throwing dice	465
8.3.3	Example: Drawing balls from a hat	468
8.3.4	Random mutations of genes	470
8.3.5	Example: Policies for limiting population growth .	475
8.4	Simple games	478
8.4.1	Guessing a number	478
8.4.2	Rolling two dice	479
8.5	Monte Carlo integration	482
8.5.1	Derivation of Monte Carlo integration	482
8.5.2	Implementation of standard Monte Carlo integration	484
8.5.3	Area computing by throwing random points	487
8.6	Random walk in one space dimension	489
8.6.1	Basic implementation	490
8.6.2	Visualization	490
8.6.3	Random walk as a difference equation	491
8.6.4	Computing statistics of the particle positions	492
8.6.5	Vectorized implementation	493
8.7	Random walk in two space dimensions	494
8.7.1	Basic implementation	495
8.7.2	Vectorized implementation	496
8.8	Summary	497
8.8.1	Chapter topics	497

8.8.2	Example: Random growth	499
8.9	Exercises	504
9	Object-oriented programming	523
9.1	Inheritance and class hierarchies	523
9.1.1	A class for straight lines	524
9.1.2	A first try on a class for parabolas	525
9.1.3	A class for parabolas using inheritance	525
9.1.4	Checking the class type	527
9.1.5	Attribute vs inheritance: has-a vs is-a relationship	528
9.1.6	Superclass for defining an interface	530
9.2	Class hierarchy for numerical differentiation	532
9.2.1	Classes for differentiation	533
9.2.2	Verification	536
9.2.3	A flexible main program	538
9.2.4	Extensions	539
9.2.5	Alternative implementation via functions	542
9.2.6	Alternative implementation via functional programming	543
9.2.7	Alternative implementation via a single class	544
9.3	Class hierarchy for numerical integration	546
9.3.1	Numerical integration methods	546
9.3.2	Classes for integration	548
9.3.3	Verification	551
9.3.4	Using the class hierarchy	552
9.3.5	About object-oriented programming	555
9.4	Class hierarchy for making drawings	556
9.4.1	Using the object collection	557
9.4.2	Example of classes for geometric objects	566
9.4.3	Adding functionality via recursion	571
9.4.4	Scaling, translating, and rotating a figure	574
9.5	Classes for DNA analysis	577
9.5.1	Class for regions	577
9.5.2	Class for genes	577
9.5.3	Subclasses	582
9.6	Summary	584
9.6.1	Chapter topics	584
9.6.2	Example: Input data reader	585
9.7	Exercises	591
A	Sequences and difference equations	601
A.1	Mathematical models based on difference equations	602
A.1.1	Interest rates	603
A.1.2	The factorial as a difference equation	606
A.1.3	Fibonacci numbers	606

- A.1.4 Growth of a population 608
 - A.1.5 Logistic growth 608
 - A.1.6 Payback of a loan 610
 - A.1.7 The integral as a difference equation 611
 - A.1.8 Taylor series as a difference equation 613
 - A.1.9 Making a living from a fortune 615
 - A.1.10 Newton’s method 615
 - A.1.11 The inverse of a function 619
- A.2 Programming with sound 621
 - A.2.1 Writing sound to file 622
 - A.2.2 Reading sound from file 622
 - A.2.3 Playing many notes 623
 - A.2.4 Music of a sequence 624
- A.3 Exercises 627
- B Introduction to discrete calculus 639**
 - B.1 Discrete functions 639
 - B.1.1 The sine function 640
 - B.1.2 Interpolation 641
 - B.1.3 Evaluating the approximation 642
 - B.1.4 Generalization 643
 - B.2 Differentiation becomes finite differences 644
 - B.2.1 Differentiating the sine function 645
 - B.2.2 Differences on a mesh 646
 - B.2.3 Generalization 648
 - B.3 Integration becomes summation 649
 - B.3.1 Dividing into subintervals 649
 - B.3.2 Integration on subintervals 651
 - B.3.3 Adding the subintervals 652
 - B.3.4 Generalization 652
 - B.4 Taylor series 654
 - B.4.1 Approximating functions close to one point 654
 - B.4.2 Approximating the exponential function 655
 - B.4.3 More accurate expansions 655
 - B.4.4 Accuracy of the approximation 657
 - B.4.5 Derivatives revisited 659
 - B.4.6 More accurate difference approximations 660
 - B.4.7 Second-order derivatives 662
 - B.5 Exercises 664
- C Introduction to differential equations 669**
 - C.1 The simplest case 670
 - C.2 Exponential growth 673
 - C.3 Logistic growth 677
 - C.4 A simple pendulum 678

C.5	A model for the spread of a disease	681
C.6	Exercises	683
D	A complete differential equation project	685
D.1	About the problem: motion and forces in physics	685
D.1.1	The physical problem	685
D.1.2	The computational algorithm	688
D.1.3	Derivation of the mathematical model	688
D.1.4	Derivation of the algorithm	690
D.2	Program development and testing	692
D.2.1	Implementation	692
D.2.2	Callback functionality	694
D.2.3	Making a module	697
D.2.4	Verification	697
D.3	Visualization	700
D.3.1	Simultaneous computation and plotting	700
D.3.2	Some applications	703
D.3.3	Remark on choosing Δt	703
D.3.4	Comparing several quantities in subplots	704
D.3.5	Comparing approximate and exact solutions	705
D.3.6	Evolution of the error as Δt decreases	706
D.4	Exercises	710
E	Programming of differential equations	711
E.1	Scalar ordinary differential equations	712
E.1.1	Examples on right-hand-side functions	712
E.1.2	The Forward Euler scheme	714
E.1.3	Function implementation	715
E.1.4	Verifying the implementation	716
E.1.5	From discrete to continuous solution	718
E.1.6	Switching numerical method	718
E.1.7	Class implementation	719
E.1.8	Logistic growth via a function-based approach ...	723
E.1.9	Logistic growth via a class-based approach	724
E.2	Systems of ordinary differential equations	727
E.2.1	Mathematical problem	727
E.2.2	Example of a system of ODEs	728
E.2.3	Function implementation	729
E.2.4	Class implementation	732
E.3	The ODESolver class hierarchy	733
E.3.1	Numerical methods	733
E.3.2	Construction of a solver hierarchy	735
E.3.3	The Backward Euler method	738
E.3.4	Verification	740
E.3.5	Example: Exponential decay	742

E.3.6	Example: The logistic equation with problem and solver classes	744
E.3.7	Example: An oscillating system	752
E.3.8	Application 4: the trajectory of a ball	754
E.3.9	Further developments of ODESolver	756
E.4	Exercises	756
F	Debugging	791
F.1	Using a debugger	791
F.2	How to debug	794
F.2.1	A recipe for program writing and debugging	794
F.2.2	Application of the recipe	797
F.2.3	Getting help from a code analyzer	809
G	Migrating Python to compiled code	811
G.1	Pure Python code for Monte Carlo simulation	812
G.1.1	The computational problem	812
G.1.2	A scalar Python implementation	812
G.1.3	A vectorized Python implementation	813
G.2	Migrating scalar Python code to Cython	815
G.2.1	A plain Cython implementation	815
G.2.2	A better Cython implementation	817
G.3	Migrating code to C	819
G.3.1	Writing a C program	819
G.3.2	Migrating loops to C code via F2PY	820
G.3.3	Migrating loops to C code via Cython	822
G.3.4	Comparing efficiency	822
H	Technical topics	825
H.1	Getting access to Python	825
H.1.1	Required software	825
H.1.2	Installing software on your laptop: Mac OS X and Windows	826
H.1.3	Anaconda and Spyder	827
H.1.4	VMWare Fusion virtual machine	828
H.1.5	Wubi for dual boot on Windows	831
H.1.6	Vagrant virtual machine	831
H.1.7	How to write and run a Python program	831
H.1.8	The SageMathCloud and Wakari web services ...	834
H.1.9	Writing IPython notebooks	835
H.2	Different ways of running Python programs	837
H.2.1	Executing Python programs in iPython	837
H.2.2	Executing Python programs in Unix	838
H.2.3	Executing Python programs in Windows	839
H.2.4	Executing Python programs in Mac OS X	840
H.2.5	Making a complete stand-alone executable	841

- H.3 Doing operating system tasks in Python 841
- H.4 Variable number of function arguments 844
 - H.4.1 Variable number of positional arguments 845
 - H.4.2 Variable number of keyword arguments 847
- H.5 Evaluating program efficiency 849
 - H.5.1 Making time measurements 849
 - H.5.2 Profiling Python programs 851
- H.6 Software testing 852
 - H.6.1 Requirements of the test function 852
 - H.6.2 Writing the test function; precomputed data 853
 - H.6.3 Writing the test function; exact numerical solution 854
 - H.6.4 Testing of function robustness 855
 - H.6.5 Automatic execution of tests 857
- References 859**
- Index 861**